第5章

統合開発環境で設計から検証までを体験する

―Active-HDL活用チュートリアル

藤永康博

ここでは，米国Aldec社のFPGA総合開発環境「Active-HDL 6.3 Student Edition」を使ってHDLによる設計とシミュレーションによる検証を体験します．サンプル回路のカウントダウン・タイマを例に，HDLソース・コードの記述からシミュレーションによる検証について具体的に解説します．　　　（筆者）

今回のサンプルは，2けたの入力を持つカウントダウン・タイマです（**図1**）．テンキーで入力する時間をセットします．［Start］ボタンを押すとカウントダウンを開始します．カウント値が0になるとブザーが鳴ります．ブザーを止める時には［Stop］ボタンを押します．HDLで記述した回路が正しく動作しているかどうかをHDLシミュレータの「Active-HDL」を用いて検証します．

1．ソース・コードの記述から検証までの流れ

図2に今回作成する回路のブロック図を示します．convブロック，controlブロック，conutブロックの三つのブ

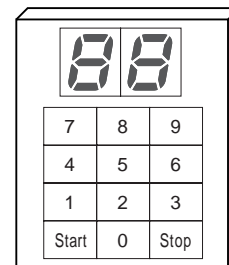



図1　カウントダウン・タイマのイメージ

2けたの入力を持ち，テンキーで入力する時間をセットする．［Start］ボタンを押すとカウントダウンを開始する．カウント値が0になるとブザーが鳴る．ブザーを止める時には［Stop］ボタンを押す．

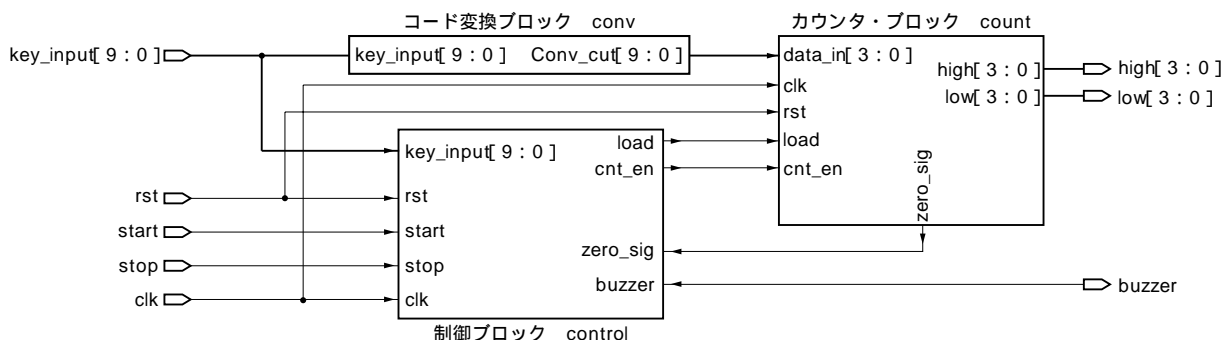



図2　カウントダウン・タイマのブロック図

三つのブロックで構成されているとてもシンプルな回路である．

KeyWord　Active-HDL，カウントダウン・タイマ，IPコア，ワークスペース，入力パターン

ロックで構成されているとてもシンプルな回路です．

convブロックは，10ビットのワンホット値を4ビットのバイナリ値に変換します．countブロックは，入力された値のカウントダウンを行います．controlブロックは，全体の制御ブロックです．

● ボトムアップ方式で設計する

今回は，ボトムアップ方式で回路の設計を進めます（**図3**）．まず，三つのコンポーネント・ブロックを作成します．

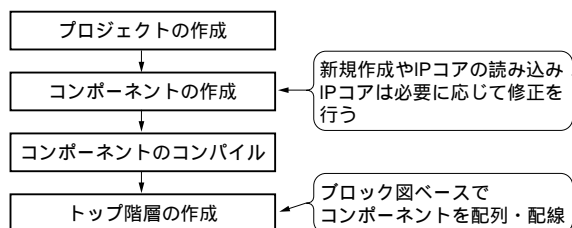



図3　カウントダウン・タイマの設計手順
今回は，ボトムアップ方式で回路の設計を進める．

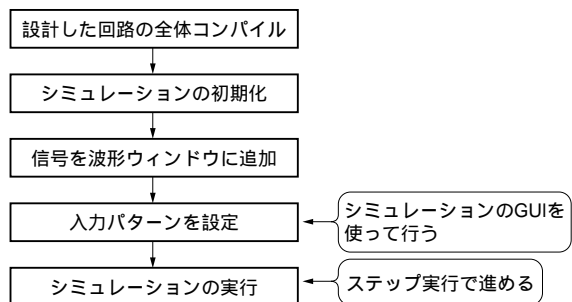



図4　カウントダウン・タイマの検証手順
入力パターンは，GUI（graphical user interface）を使って作成する．

その後，それらのコンポーネントを使ってカウントダウン・タイマ回路（トップ階層）を作成します．

countブロックとcontrolブロックについては，あらかじめ設計済みのIP（intellectual property）コアを使用します．これは付属DVD-ROMに収録されています（**表1**）．convブロックは新しく作成します．

● シミュレーションで検証する

設計した回路のシミュレーションを行います．今回はGUI（graphical user interface）を使って入力パターンを与えます．入力パターンが簡単なときには，GUIを使うと手軽にシミュレーションできて便利です．検証の手順を**図4**に示します．

2．カウントダウン・タイマの設計

Windowsのデスクトップにあるアイコンをダブル・クリックして，Active-HDLを起動します．

表1　あらかじめ設計済みの機能ブロック

ファイル名	説　明
bcdcounter.vhd	カウンタ・ブロックを構成するVHDLファイル．
count.bde	カウンタ・ブロック本体．Active-HDLのブロック・ダイヤグラム・エディタで作成したもの．bcdconter.vhdを下位階層に持つ．
control.asf	制御ブロック．Active-HDLのステート・マシン・エディタで作成した．

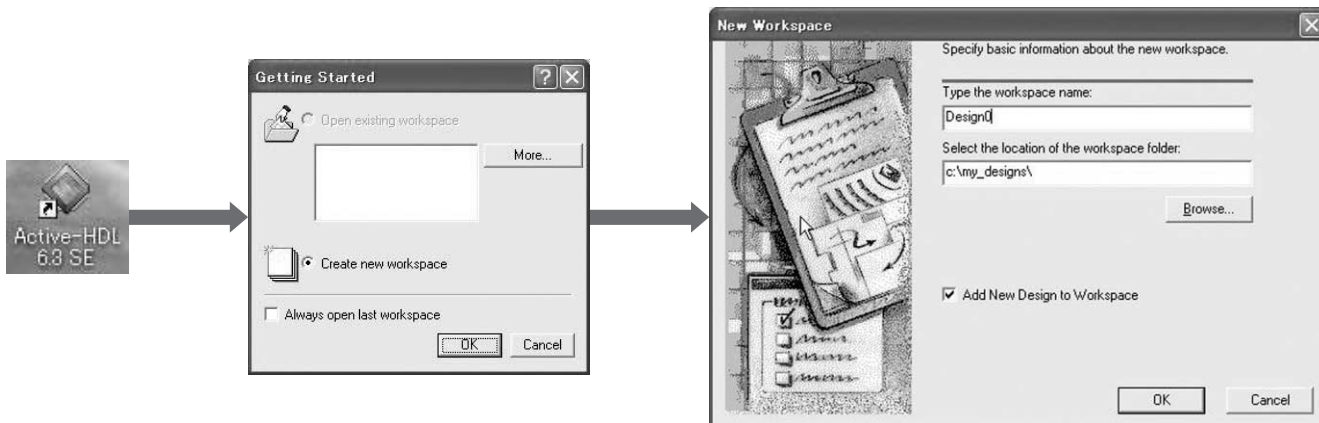
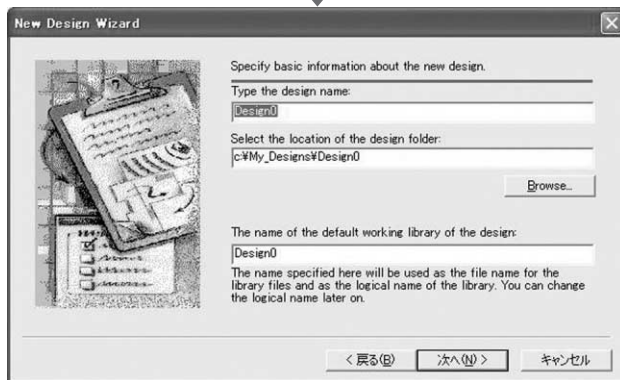
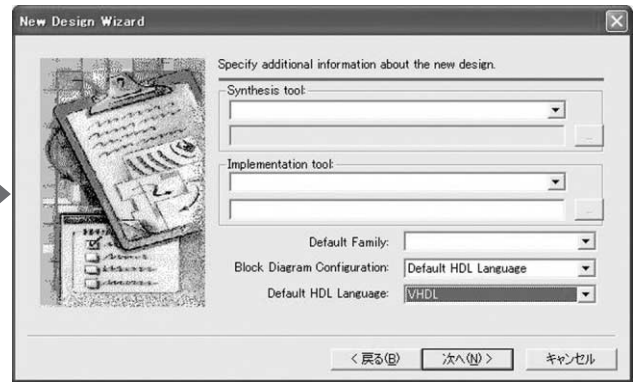
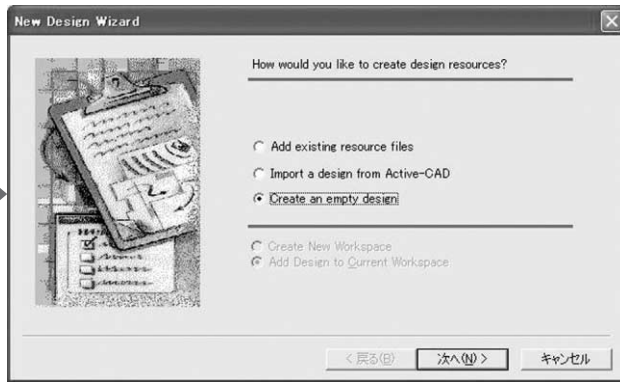



図5　プロジェクト（ワークスペース）の作成
Active-HDLでは，プロジェクトのことをワークスペース（workspace）と呼んでいる．

メニュー
ウィンドウの起動やシミュレーションなどの操作が可能

デザイン・ブラウザ
デザインのソースや各種ファイルの管理

作業スペース
各種エディタや，波形ウィンドウなどを表示

コンソール
実行ログの表示やコマンド入力

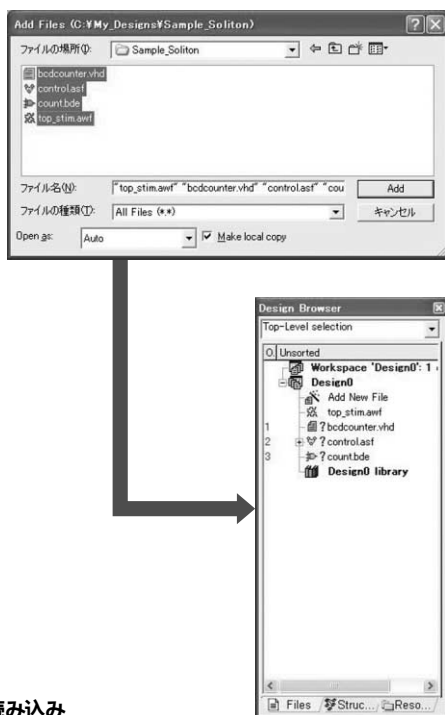



図6
コンポーネントの読み込み

● **プロジェクト（ワークスペース）の作成**

プロジェクトの作成手順を**図5**に示します．

1）ワークスペースの新規作成

Active-HDLを起動すると，Getting Startedウィンドウが表示されます．プロジェクトを新しく作成したいので，「Create new workspace」をチェックし，［OK］ボタンをクリックします．

最近は，「プロジェクト」という単位で設計を管理するツールが多いようです．Active-HDLでも同様のしくみで設計を管理しますが，プロジェクトのことを「ワークスペース（workspace）」と呼んでいます．ワークスペースは，プロジェクトと同じようなものと理解してください．

なお，すでに作成済みのワークスペースがある場合は，Open existing workspaceのボックスにリスト表示されます．ワークスペースを選択して［OK］ボタンを押すと，そのワークスペースが開きます．

2）ワースペースの設定

New Workspaceウィンドウでは，作成するワークスペースの名前を入力します．今回は「Design0」とします注1．

注1：付属DVD-ROMに収録されているActive-HDL Student Editionは，ワークスペース名として「Design0」から「Design10」までのいずれかの名称しか使用できない制限がある．

「Add New Design to Workspace」をチェックして［OK］ボタンをクリックします．

Select the location of the workspace folderにはインストール時に設定したパス（デフォルトならc:\my_designs\）が表示されています．ワークスペースのフォルダは，my_designの直下にDesign0という名称で作成されます．

3）デザインの新規作成

Active-HDLでは，ワークスペースに「デザイン（design）」を作成し，そこでHDLコードの作成や既存のHDLコードの読み込みを行って，設計を進めていきます．

New Design Wizardでは，このデザインの設定を行います．今回は新しいデザインを作成するため，ウィザードの1ページ目では「Create an empty design」にチェックして［次へ］ボタンをクリックします．

4）デザイン情報の入力

ウィザードの2ページ目では，作成するデザインに関する情報を入力します．今回はDefault HDL Languageで「VHDL」を選択します．ほかの項目はブランクのまま，［次へ］をクリックします注2．

5）デザイン名の入力

ウィザードの3ページ目では，デザインの名前を入力します．今回は，Type the design nameとして「Design0」と入力します注3．The name of the default working library of the designのボックスにはデザインと同じ名称が自動的に入力されます．

6）デザインの確認

ウィザードの4ページ目では，設定したデザイン名を確認します．［完了］ボタンをクリックすれば，ワークスペースとデザインの作成が完了します．

7）メイン画面の表示

ワークスペースとデザインの作成が終了するとActive-HDLのメイン画面が表示されます．Active-HDLの画面は，メニュー，デザイン・ブラウザ，作業スペース，コンソールで構成されています．

● **コンポーネントの作成（読み込み）**

カウントダウン・タイマを構成する三つのコンポーネン

注2：他社の論理合成ツール，配置配線ツールを使用する場合は，この画面で選択すれば，Acitve-HDLにプラグインして使用できる．プラグインできるツールは「Synthesis tool」，「Implementation tool」のプルダウンで確認できる．

注3：デザイン名もワークスペースと同様に「Design0」から「Design10」までの名称しか使用できない．

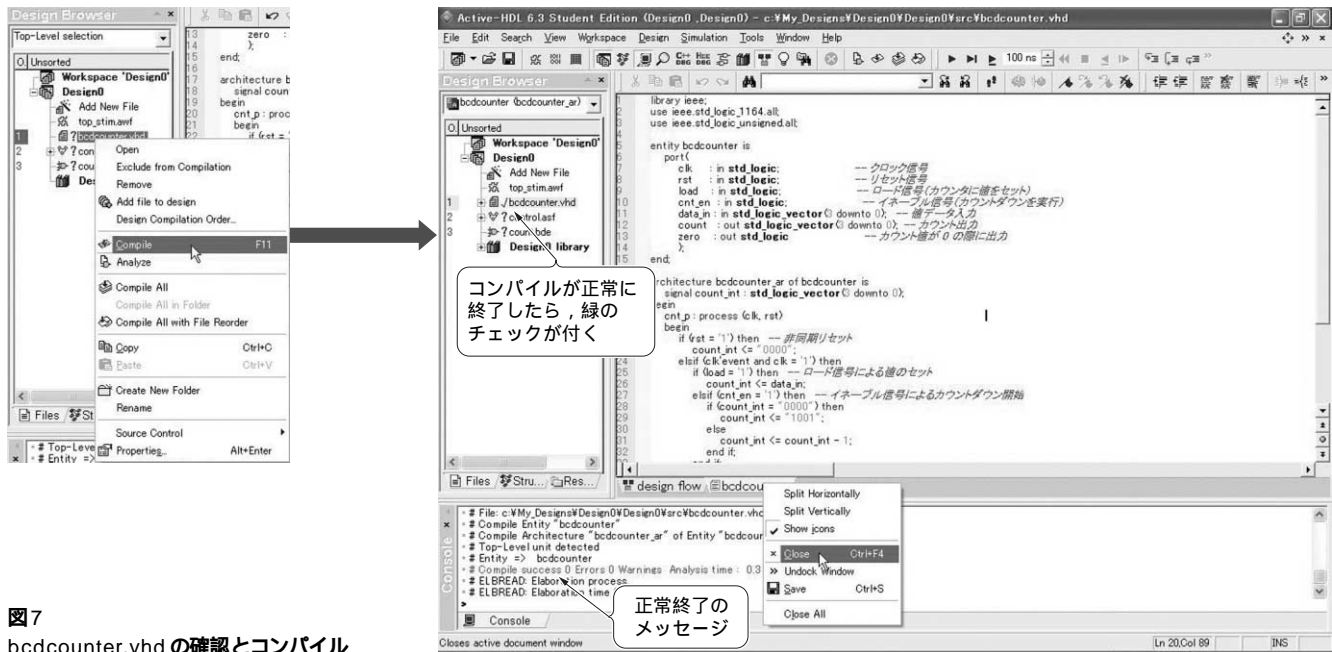



図7
bcdcounter.vhdの確認とコンパイル

トを作成します．今回は付属DVD-ROMに収録済みのファイルを使用します．エディタなどを使ってHDLソース・コードを作成した後をイメージしてください．

1）コンポーネントの読み込み

コンポーネントの読み込みは，メニューから「Design」→「Add Files to Design」で行います．

図6のように，すべてのファイルを選択し，Make local copyにチェックが入っていることを確認して，［Add］ボタンをクリックします．すると，デザイン・ブラウザにファイルが登録されます．

2）bcdcounter.vhdの確認とコンパイル

読み込んだファイルの内容確認とコンパイルを行います．

デザイン・ブラウザで「bcdcounter.vhd」をダブル・クリックします．すると，作業スペースにVHDLソースコードが表示されます．Active-HDLのテキスト・エディタでは，HDL，C，Tclなどの構文をカラー表示します．

デザイン・ブラウザのFilesタブで，「bcdcounter.vhd」を右クリックして表示されるメニューから「Compile」を選択します（**図7**）．コンパイルが正常に終了したら，コンソールにメッセージ（Compile success）が出力され，ファイル名の左横に緑のチェックが付きます．

作業スペースのbcdcounter.vhdタブを右クリックして表示されるメニューから「Close」を選択し，ウィンドウを閉じます．

3）count.bdeの確認とコンパイル

次に，デザイン・ブラウザで「count.bde」をダブル・クリックします．すると，作業スペースにブロック・ダイヤグラムが表示されます（**図8**）．

このブロックはbcdcounter.vhdをコンポーネントとして作成されています．U1とU2のブロックをダブル・クリックすると，ブロックの内容であるbcdcounter.vhdのコードが作業スペースに表示されます．

デザイン・ブラウザのcount.bdeタブを右クリックし，表示されるメニューから「Compile」を選択してコンパイルを行います．コンパイルが正常に終了すれば，VHDLコードとシミュレーション・ライブラリが生成されます．

デザイン・ブラウザでcount.bdeを展開すると，count.vhdとcount（count）が生成されていることが分かります．「count.vhd」をダブル・クリックすると作業スペースにソース・コードが表示されます．

4）control.asfの確認とコンパイル

デザイン・ブラウザで「control.asf」をダブル・クリックすると，作業スペースにステート・マシン・エディタが表示されます（**図9**）．上の方に並んでいるのが入出力ポートです．Sreg0で囲まれた枠内にステート・マシンが記述されています．

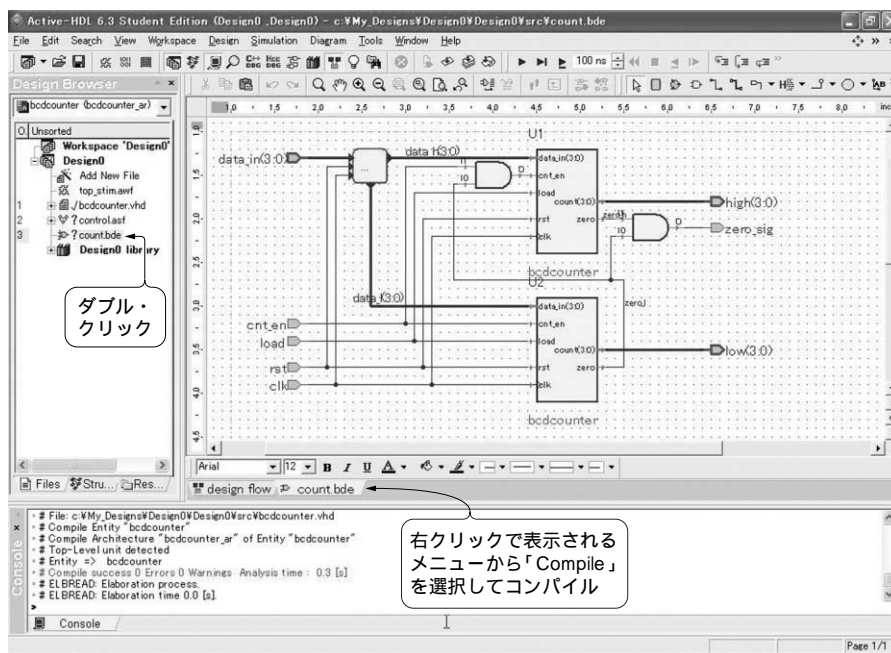



図8
count.bde の確認とコンパイル

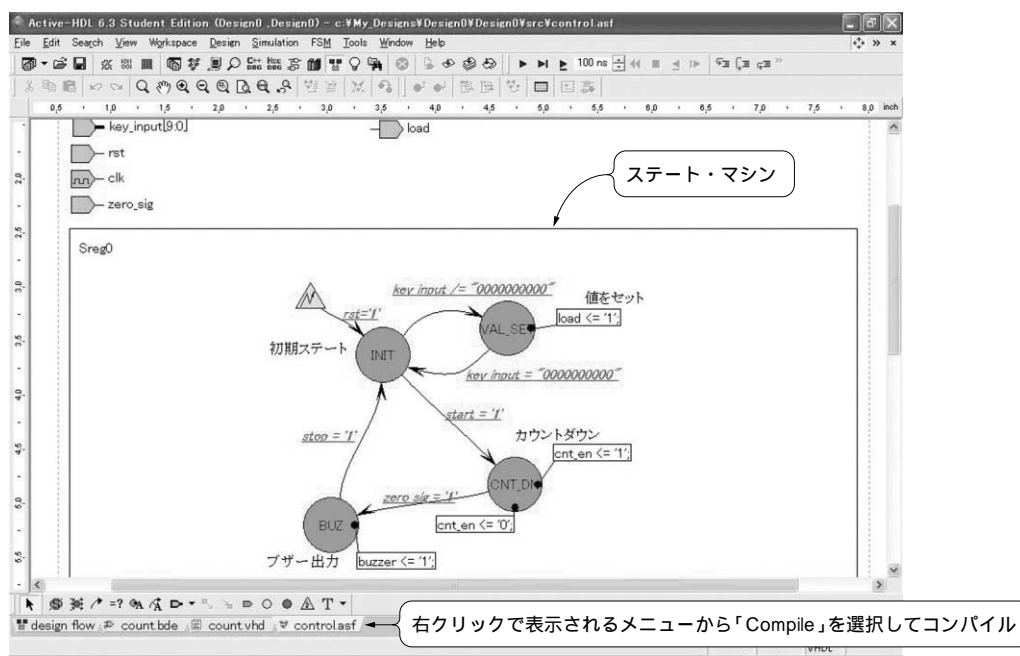



図9
control.asf の確認とコンパイル

デザイン・ブラウザのcontrol.asfタブを右クリックして，表示されるメニューから「Compile」を選択し，コンパイルを行います．コンパイルが正常に終了すると，control.asfの下にcontrol.vhdとcontrol（control）が生成されます．

5）conv ブロックの作成

convブロックを作成します．今回は，IPコア・ジェネレータという機能を使用します．メニューから「Tools」→「IP CORE Cenerator」を選択します．

コード変換を行う回路を生成するCode Convertersを使用します（**図10**）．IPコア・ジェネレータの画面から「Code Converters」をクリックすると，作業スペースに入力画面が表示されるので，10ビットの1ホット（“1”アクティブ）信号をバイナリ値に変換する回路になるように設定します．回路が生成されると，デザイン・ブラウザにip_coreフォルダとconvフォルダが追加されます．convフォルダを展開すると，三つのファイルが格納されています（**表2**）．

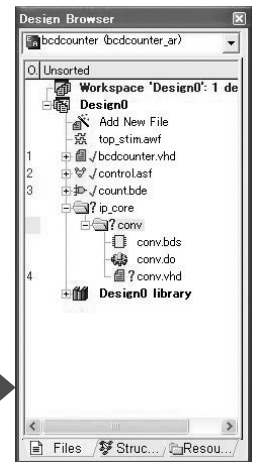
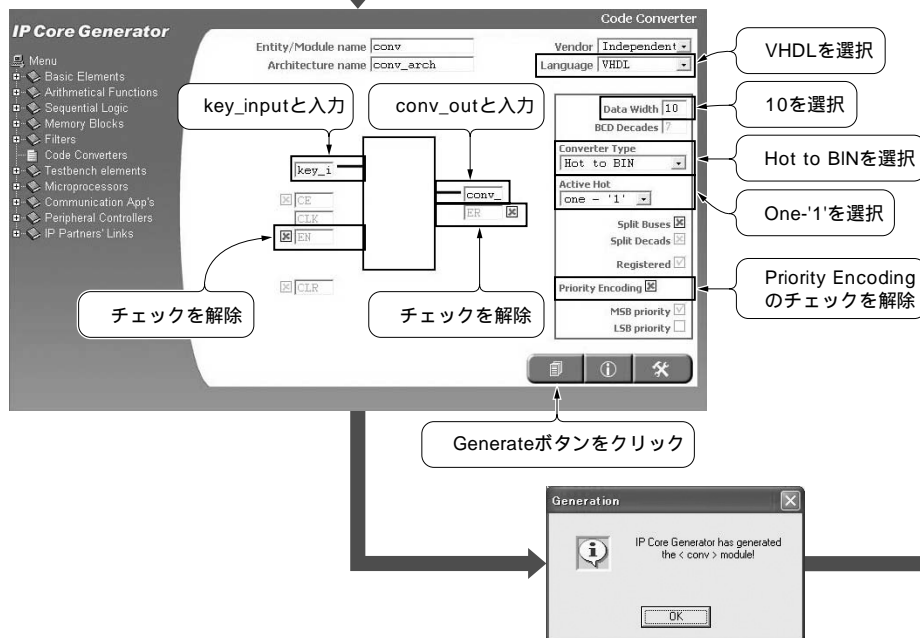
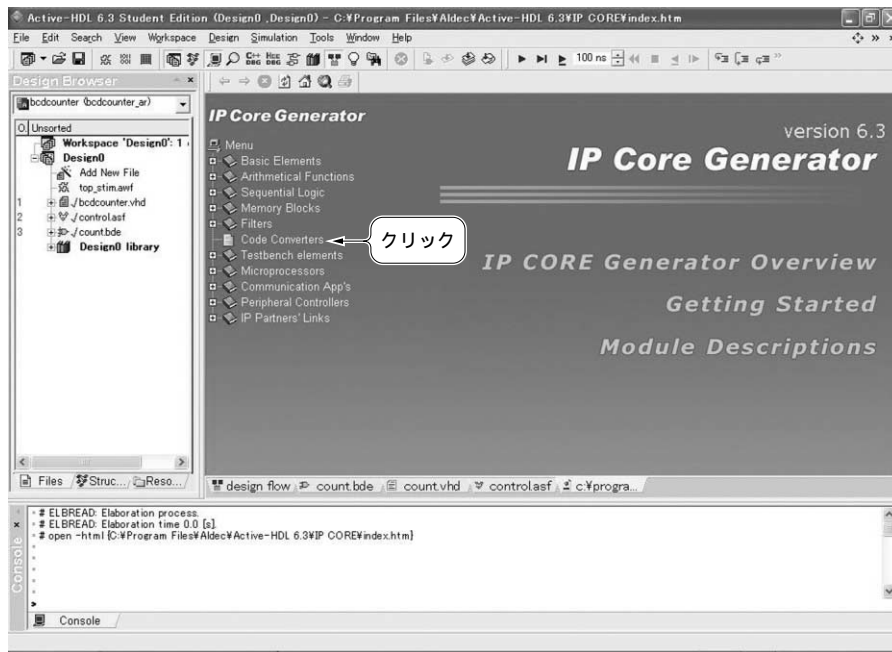



図10　conv ブロックの作成
IP コア・ジェネレータにより，コード変換を行う回路を生成する．回路の仕様を GUI で入力すると，自動的に生成される．

作業スペースのIPコア・ジェネレータ・タブ（C:\progra...と表記）を右クリックして，表示されるメニューから「Close」を選択して，IPコア・ジェネレータを終了します．

6）自動生成されたHDLコードの修正とコンパイル

デザイン・ブラウザで「conv.vhd」をダブル・クリックして，作業スペースにソース・コードを表示します．自動生

表2　IP コア・ジェネレータにより生成されるファイル

ファイル	説　明
conv.bds	生成されたVHDLファイルに相当するシンボル．ブロック・ダイヤグラムで使用可能
conv.do	生成されたVHDLファイルをコンパイルするためのマクロ・ファイル
conv.vhd	生成されたVHDLファイル

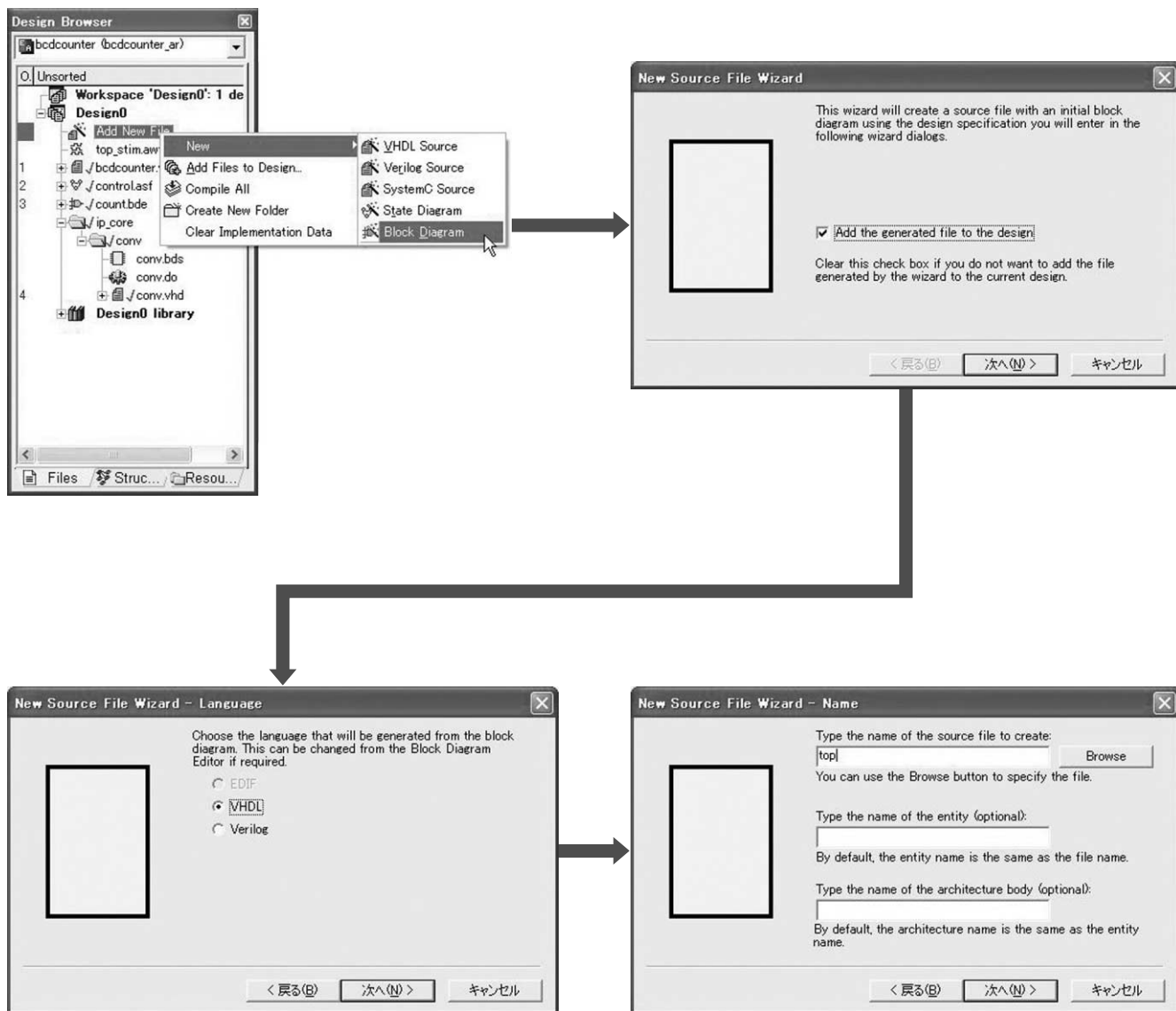



図11　新規ブロック・ダイヤグラムの作成

リスト1　自動生成されたHDLコードの修正

```
修正前    when others => conv_out <= (others => '0');

修正後    when others => conv_out <= "1111";
```

成されたコードは，完全に期待通りのものとは限りません．今回は，1ホット以外の値が入った場合に無効な値として処理させるために，49行目を**リスト1**のように修正します．

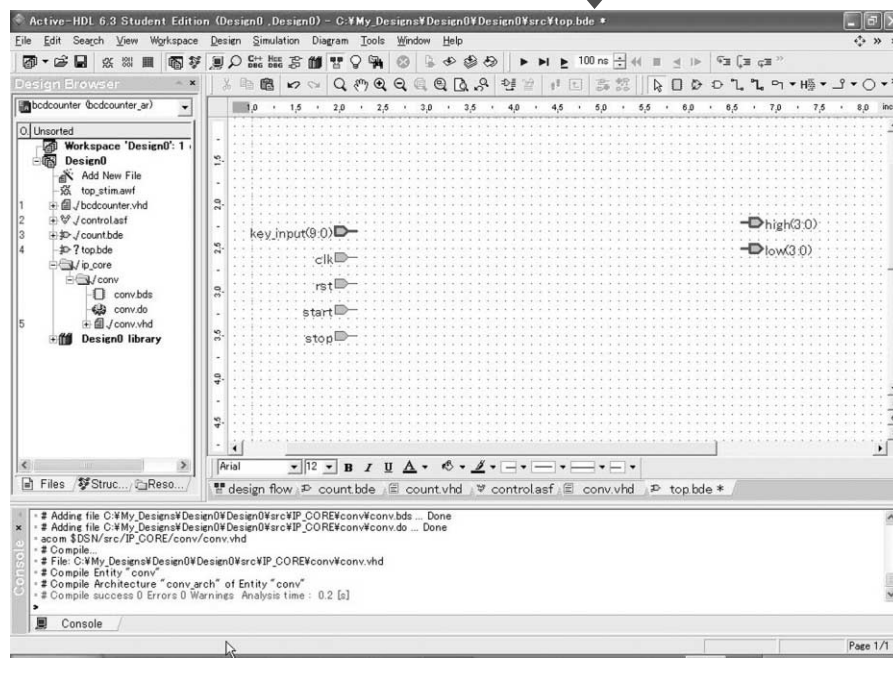
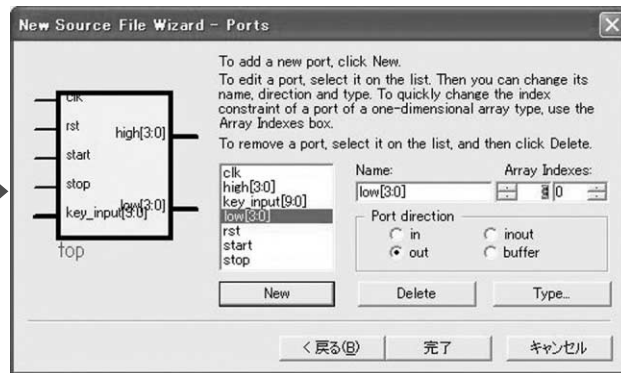
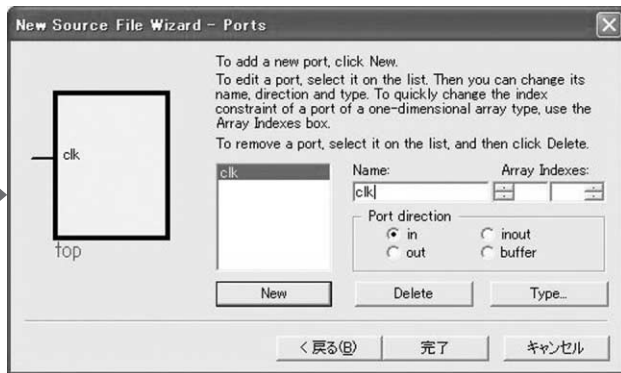
「conv.vhd」を右クリックして，表示されるメニューから「Compile」を選択し，修正したconv.vhdをコンパイルします．

● トップ階層の作成

ブロック・ダイヤグラムで三つのコンポーネントを配置・配線し，トップ階層を作成します．

1) 新規ブロック・ダイヤグラムの作成

デザイン・ブラウザの「Add New File」を右クリックして，表示されるメニューから「New」→「Block Diagram」を選択して，ブロック・ダイヤグラムを新規に作成します（**図11**）．すると，New Source File Wizardが起動します．

ウィザードの1ページ目では，設定後に生成されるファイルをデザイン・ブラウザ内に追加する指定を行います．「Add the generated file to design」にチェックを入れ，［次へ］ボタンを押します．

ウィザードの2ページ目では，生成する設計言語を選択します．「VHDL」をチェックして［次へ］ボタンを押します．

ウィザードの3ページ目では，ブロック・ダイヤグラムの名前と，Entity名，Architecture名を入力します．Entity名とArchitecture名を入力しない場合は，ブロック・ダイヤグラムと同じ名前で生成されます．ここではブロック・ダイヤグラムの名前として「top」だけを入力し，［次へ］ボタンを押します．

ウィザードの4ページ目では，トップ階層の入出力ポートを設定します．［New］ボタンを押し，Name欄にポート名として「clk」と入力します．入力ポートであれば，Port directionで「in」をチェックします．これで画面左のブロックにclkが追加されます．lowのようなバスを指定するときには，Array Indexesにバス幅を入力します．low［3：0］の4ビット・バスであれば，左の欄に「3」，右の欄に「0」を入力します．この手順を繰り返して，**図2**で示したブロック図のように，入出力ポートを配置していきます．出力ポートでは，Port directionで「out」をチェックします．入出力ポートを設定したら，［完了］ボタンを押します．

2）コンポーネントの配置

用意した三つのコンポーネントをtopブロックに配置します．配置にはSymbols Toolboxを利用します（**図12**）．

ブロック・ダイヤグラムの右に表示されるSymbols Toolboxには，デザインに含まれるコンポーネントが表示

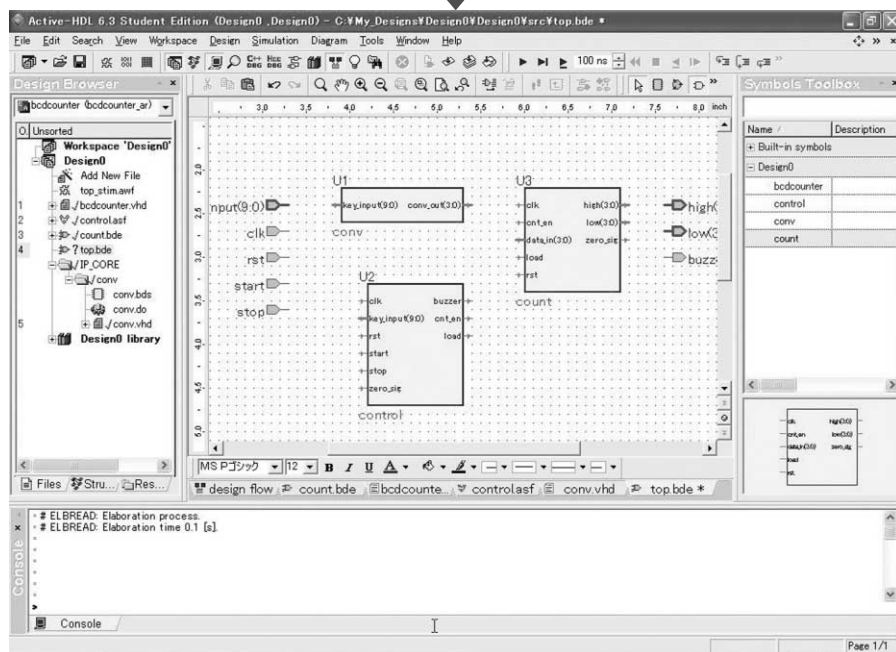



図12　コンポーネントの配置

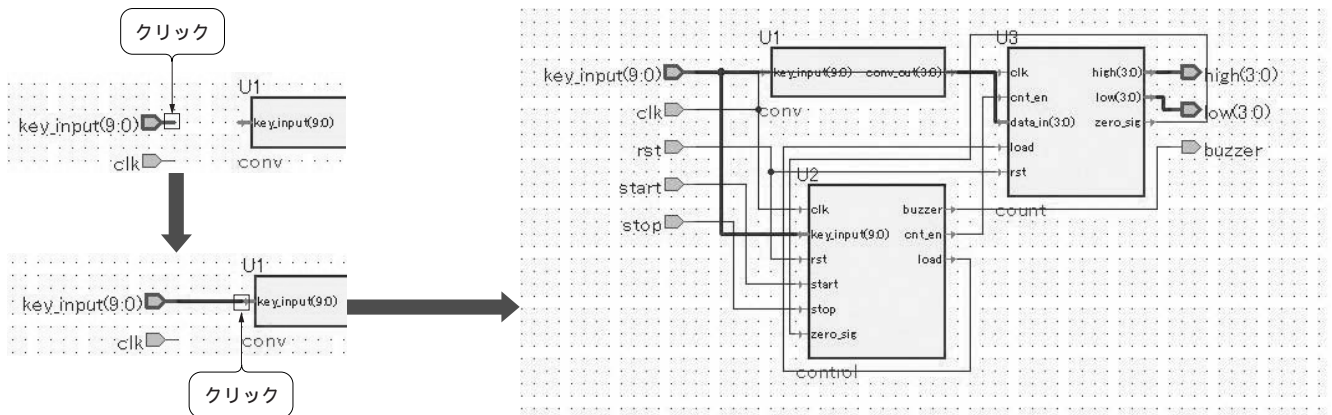



図13 コンポーネントの接続

されています．各ブロックをクリックすると，シンボルが表示されます．

Symbols Toolboxで「conv」を選択し，表示されたシンボルをブロック・ダイヤグラムへドラッグ&ドロップします．同様に「control」と「count」をSymbols Toolbox からドラッグ&ドロップします．

シンボルの配置が完了したら，Symbols Toolbox を閉じます．

3）コンポーネントの接続

配置したコンポーネント間の信号を接続します．

まず，入力ポートkey_input［9：0］とconvブロックの入力key_input［9：0］を接続します（**図13**）．マウスのカーソルを入力ポートに合わせます．カーソルが「⇒」から「＋」に変化したのを確認して，マウスを左クリックします．convブロックの入力ポートへマウスを移動し，もう一度マウスを左クリックします．これで二つの入力を接続します．

同じ要領で，すべての配線を接続します．バスとワイヤは，ポートの状態に合わせて自動認識されます．例えば，入力ポートのclkとcontrolブロックの入力clkを接続すると，ワイヤで接続されます．配線が完了したら，セーブしておきます．

3．シミュレーションによる検証

サンプル回路のHDLソース・コードの作成が終了しました．次に記述した回路が正しく動作するかどうか確認する必要があります．ここでは，シミュレーションによる検証を行います．

● デザインのコンパイル

まず，デザイン全体をコンパイルします．

メニューの［アイコン］ボタンをクリックします．Active-HDLでは階層構造を認識し，自動的に下位階層からコンパイルを実行します．VHDLのコンパイル時も階層を意識しなくても自動的にツールが判別し，コンパイルを行います．

● シミュレーションの初期化

初めに，コンパイル後に生成されたライブラリのトップ階層を指定します．デザイン・ブラウザのワーキング・ライブラリ「Design0 library」を展開し，「top（top）」を右クリックして表示されるメニューから「Set as Top-Level」を選択します．

次に，メニューの「Simulation」→「Initialize Simulation」を選択します．エラボレーションが実行され，シミュレータが起動します．シミュレータが起動するとデザイン・ブラウザが自動的にStructureタブに移動し，階層が表示されます．

● 入力パターンの設定

1）トップ階層の全信号を波形ウィンドウに追加

波形ウィンドウを起動し，観測信号を追加します（**図14**）．デザイン・ブラウザの「top（top）」を波形ウィンドウにドラッグ&ドロップすると全信号を追加できます．

波形ウィンドウで入力信号と出力信号を見やすくするため，空白の行を挿入します．波形ウィンドウの「Bus159」

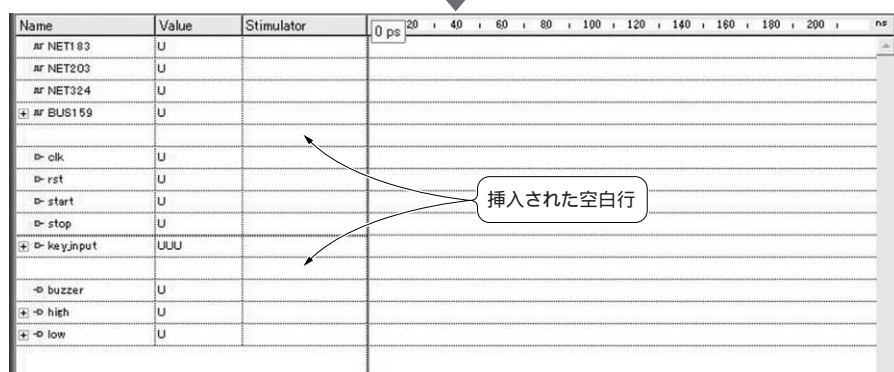
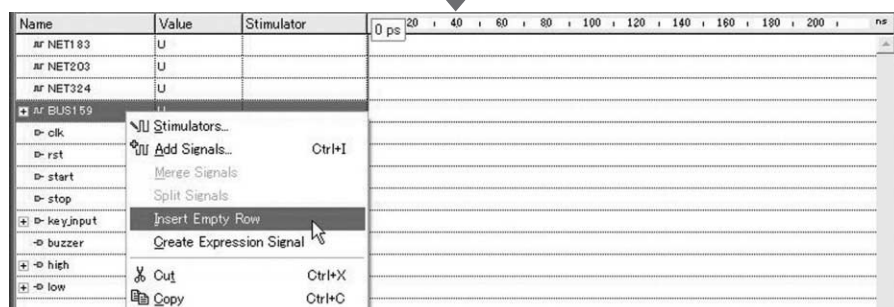
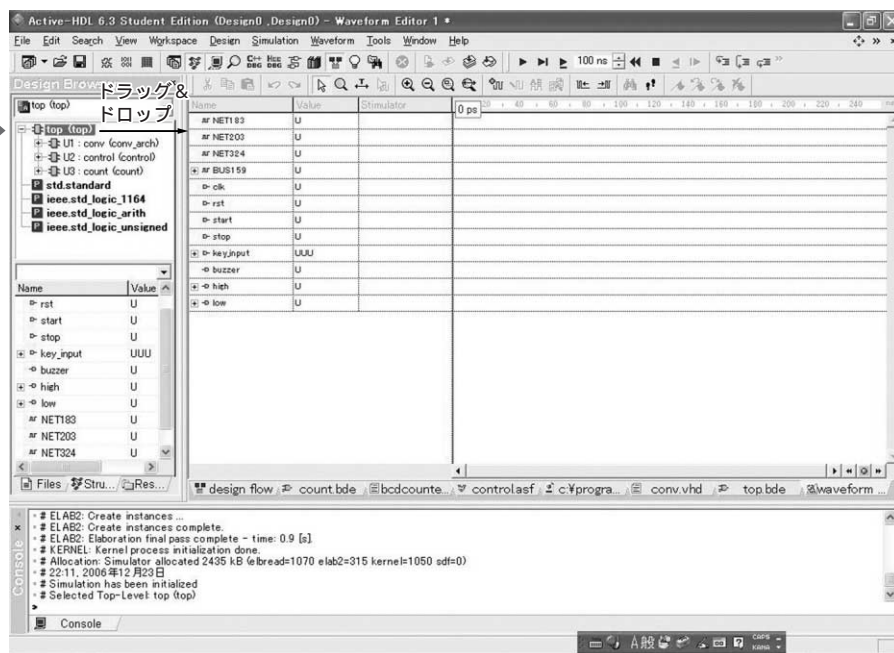



図14
トップ階層の全信号を波形ウィンドウに追加

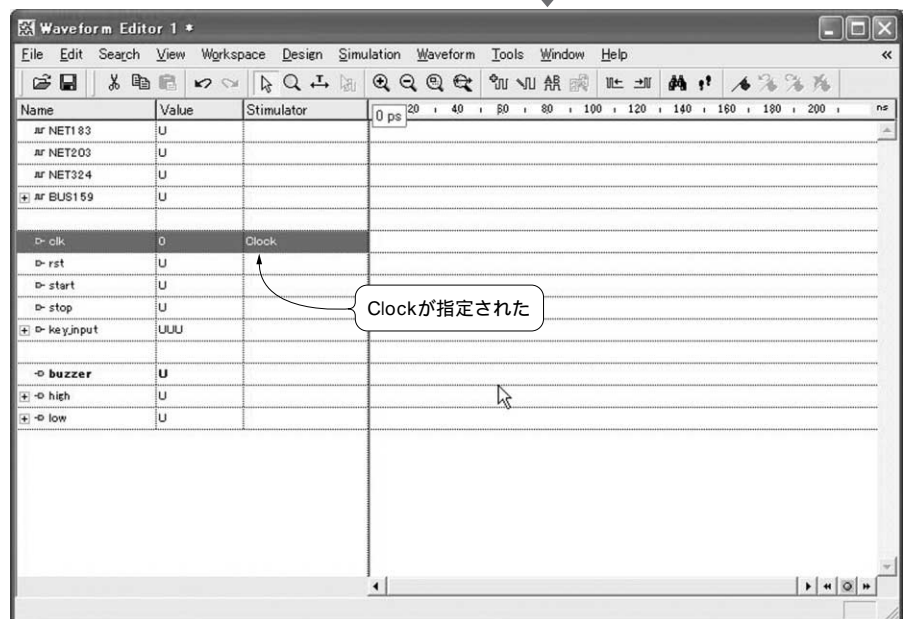
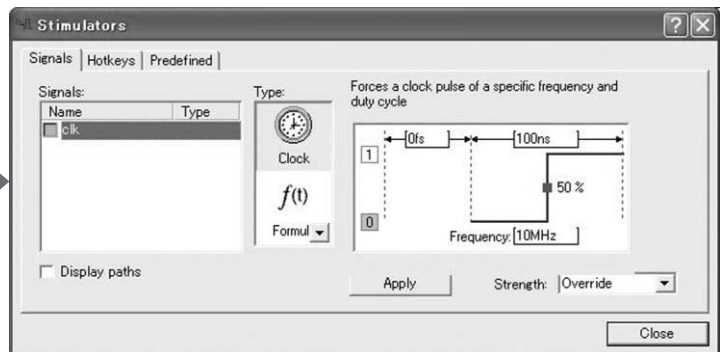



図15　clk信号の設定

を右クリックして，表示されるメニューから「Insert Empty Row」を選択します．同様に「key_input」を右クリックし，「Insert Empty Row」を選択します．

2）clk信号の設定

今回は入力パターンの設定に「Stimulators」という機能を使用します．

波形ウィンドウの「clk」を右クリックして，表示されるメニューから「Stimulators」を選択します（**図15**）．Stimulatorsウィンドウでは，Typeで「Clock」を選択すると，クロック周波数を設定する項目が表示されます．今回はデフォルトのままにします．設定が終了したら，［Apply］ボタンをクリックします．波形ウィンドウのclk行を見ると「Clock」が指定されたことを確認できます．

3）rst信号の設定

Stimulators画面を起動したまま，波形ウィンドウの「rst」をクリックすると，Stimulators画面に「rst」が追加されます．

Stimulators画面の「rst」をクリックし，「Type」で「Fomula」を選択します．**図16**に示したように入力を行って，［Apply］ボタンをクリックします．

4）start信号とstop信号の設定

シミュレーション実行中にキーボードの「Q」を押すと，入力「start」がトグルするように設定します（**図17**）．

Stimulators画面を起動したまま，波形ウィンドウの「start」をクリックすると，Stimulators画面に「start」が追加されます．Type欄では「Hotkey」を選択します．そし

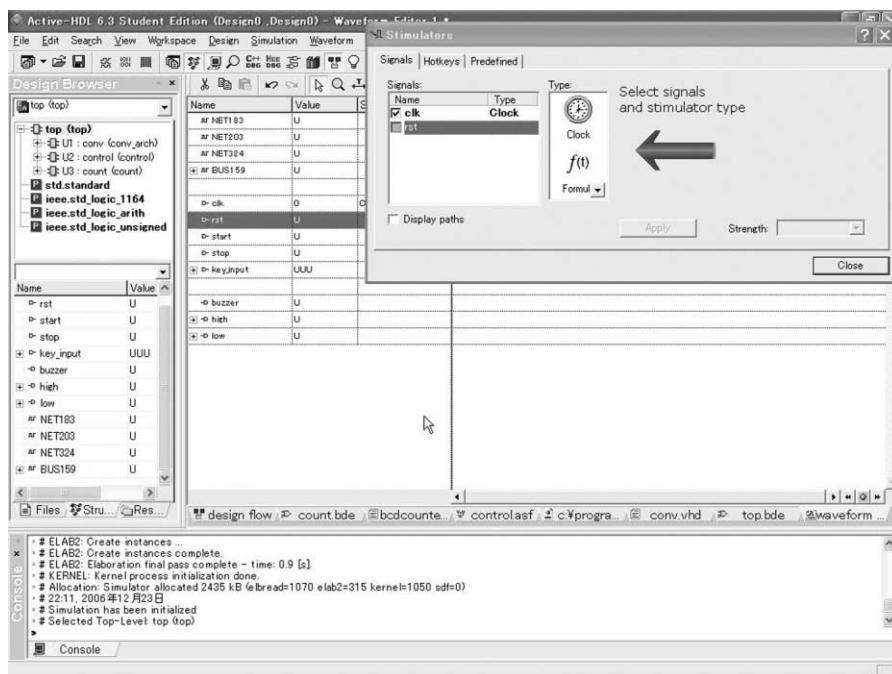



図16　rst信号の設定

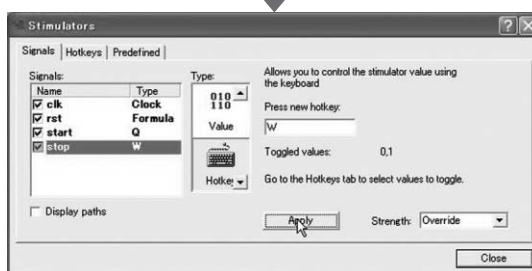
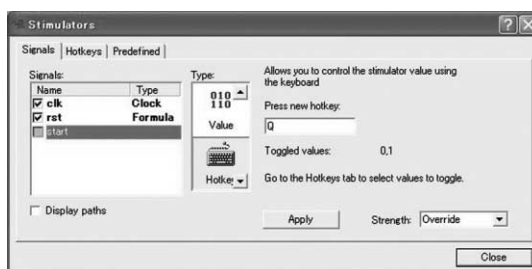
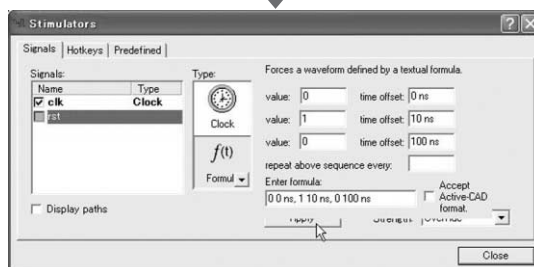



図17　start信号とstop信号の設定

Name	Value	Stimulator
NET324		
BUS159		
clk		Clock
rst		Formula
start		Q
stop		W
key_input		
key_input(9)		9
key_input(8)		8
key_input(7)		7
key_input(6)		6
key_input(5)		5
key_input(4)		4
key_input(3)		3
key_input(2)		2
key_input(1)		1
key_input(0)		0

図18　キー入力の設定

図19 ステップ実行

て，パソコンのキーボードの「Q」キーを押すと，Press new hotkey欄に「Q」が設定されます．また，Signals欄のstartにチェックが入り，Qと表示されます．

同様にstop信号も設定します．stop信号のトグルには「W」キーを使うように設定します．

5）key_input信号の設定

key_inputでは，0から9までの数をキーボードに割り付けます．波形ウィンドウの「key_input」を展開します．

後はstart信号のときと同じように，key_inputの各入力にキーを割り付けていきます．0の割り付けでは，波形ウィンドウの「key_input(0)」を選択し，Stimulatorsウィンドウで「Hotkey」を選択してから，パソコンのキーボードの「0」キーを押します．[Apply]ボタンをクリックすると，0キーの割り付けが完了します．9までの10個のキーの設定が終了したら，Stimulatorsウィンドウの[Close]ボタンをクリックします．

波形ウィンドウの入力が，**図18**のように設定されているかどうかを確認してください．

● シミュレーションの実行

今回はステップ実行を行いながら（入力信号に割り当てたキーボードを操作しながら）シミュレーションを行います．

ステップ実行を行うには，**図19**に示すボタンをクリックします．ボタン右横のボックスに1回のステップで実行される時間が示されています．また，キーボードの「F5」キーを押しても同じ操作ができます．

ボタンまたは「F5」キーを1回押すと，シミュレーションが100nsまで進みます．もう一度押すと，200nsまでの波

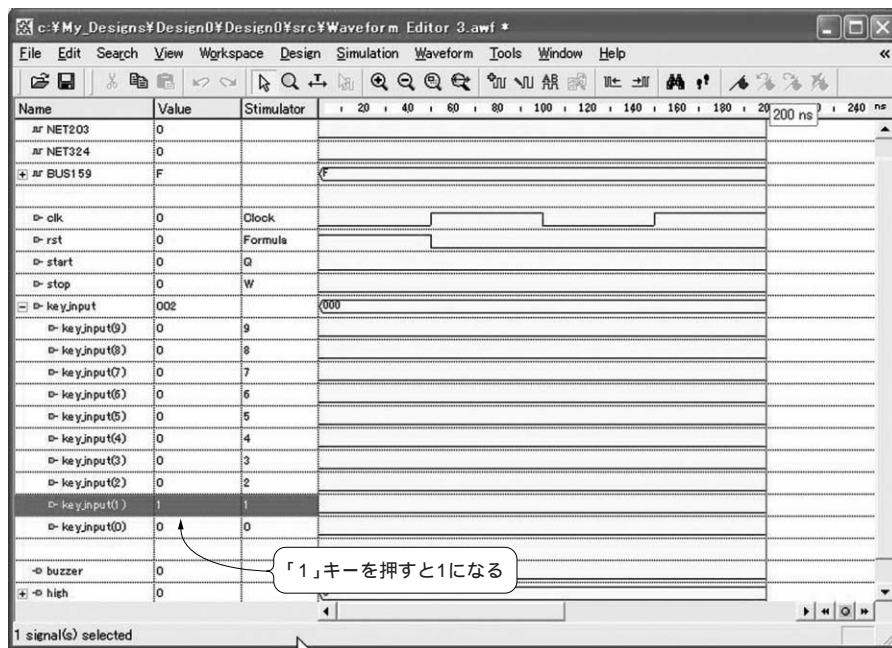
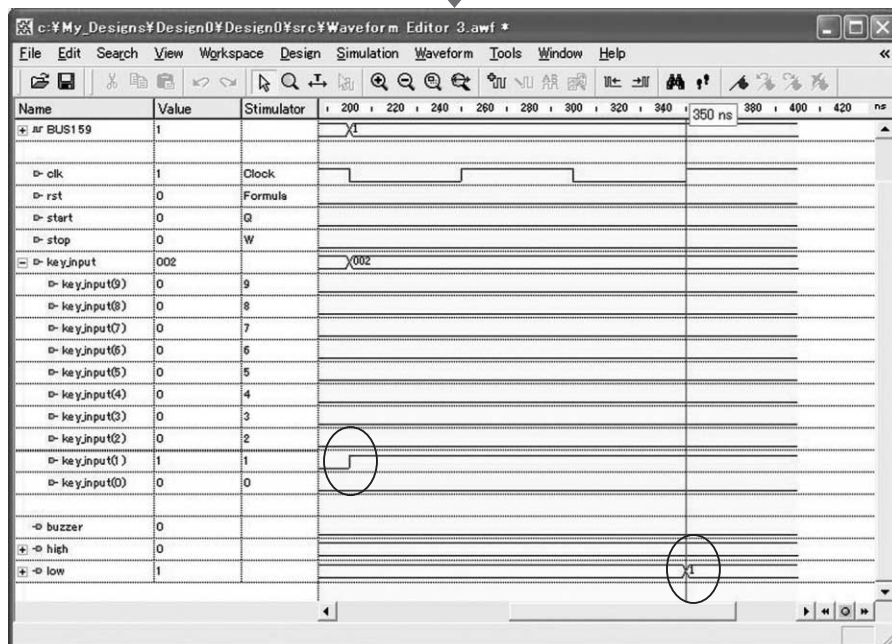



図20　時間のセットをシミュレーション

形が表示されます．

1）タイマに時間をセット

タイマに時間をセットする動作をシミュレーションします．ここでは12をセットします（**図20**）．

key_input（1）の値を0から1に変化させるために「1」キーを押します．波形ウィンドウではkey_input（1）のValueが1になります．

シミュレーションをさらに200ns進めます．350nsのクロックの立ち上がりでlowの値が1になっていることが分かります．もう一度「1」キーを押してkey_input（1）の入力値を0に戻します．

次に「2」キーを押してkey_input（2）の入力値を1に変化させます．波形ウィンドウのkey_input（1）とkey_input（2）のValueがそれぞれ0，1になっていることを確認して，シミュレーションをさらに200ns進めます．波形は600nsまで表示されています．550nsのクロックの立ち上

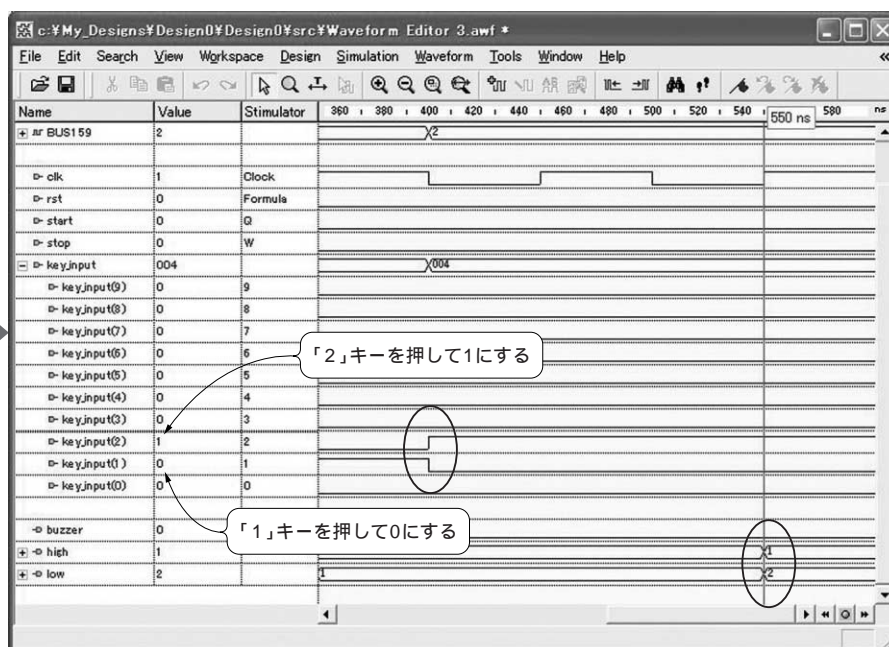
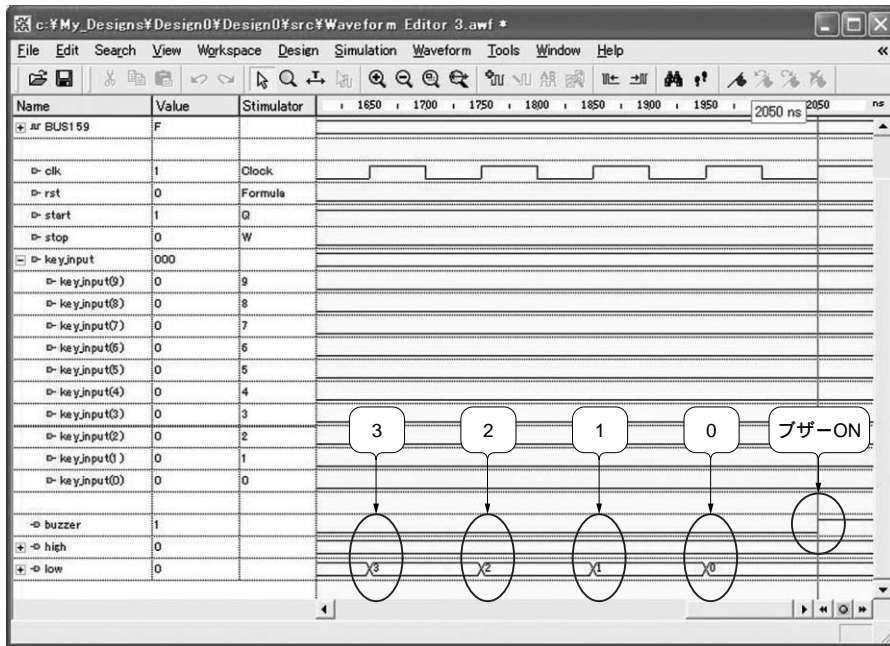



図22　ブザーの動作をシミュレーション

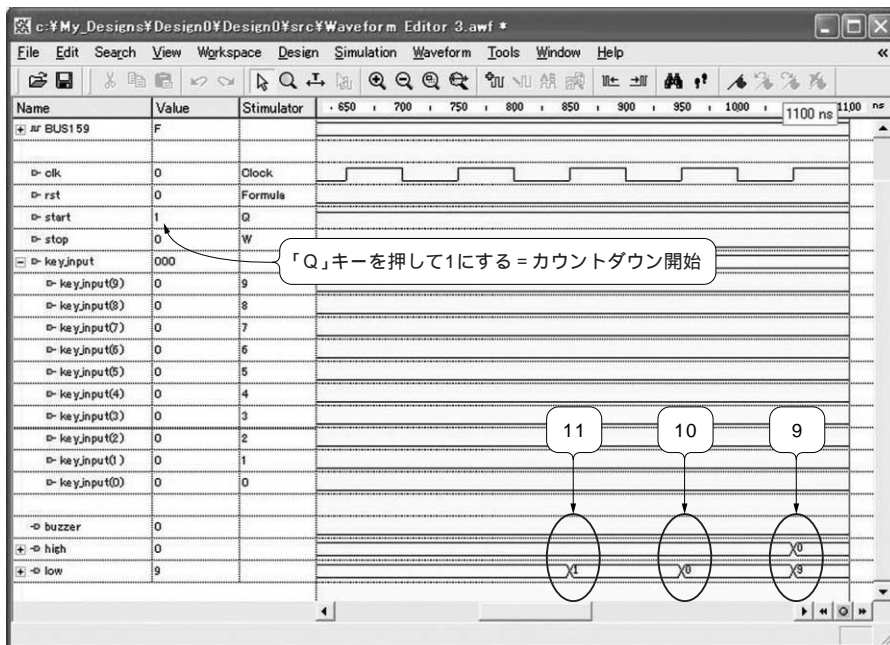
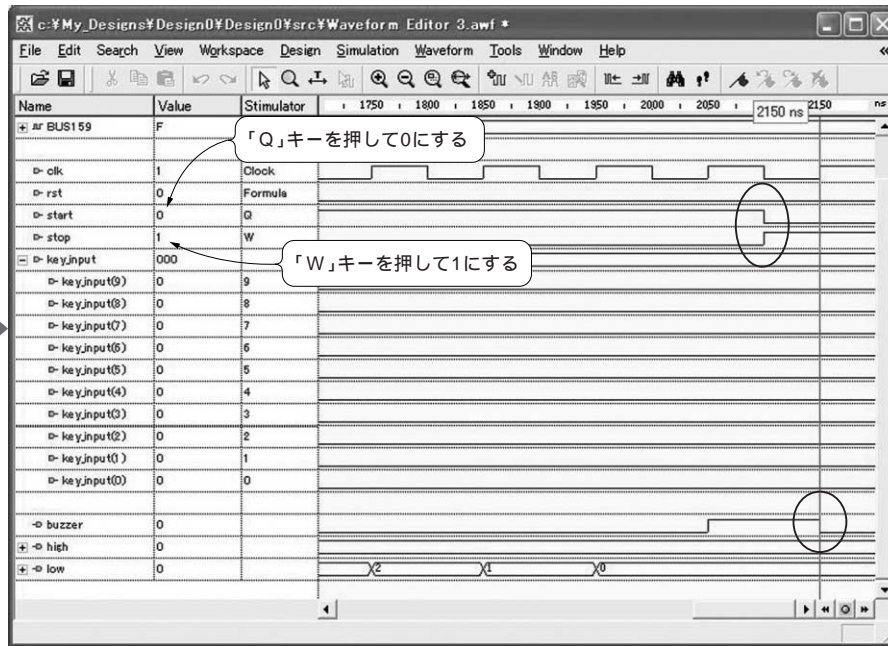



図21　カウントダウンをシミュレーション

5

がりで信号highに1が，lowに2が出力されていることが確認できます．key_input(2)の入力値を0に戻すために「2」キーを押します．

2）カウントダウンの開始

カウントダウンの動作をシミュレーションします．

Start信号の値を1に変化させるため，「Q」キーを押します．シミュレーションを進めていくとカウントダウンの様子が確認できます（**図21**）．

3）ブザーの動作

カウント値が0になるまで，シミュレーションを進めていきます．1950nsで出力が0になります．出力が0になった次のクロックの立ち上がり（2050ns）で出力信号buzzerが1に変化します（**図22**）．

「Q」キーを押してstartの値を0に戻しておきます．その後，「W」キーを押してstopを1にします．

シミュレーションを進めると，stop信号が1になった次のクロックの立ち上がりでbuzzerが0に変化します．

ふじなが・やすひろ
(株)ソリトンシステムズ